43rd AIAA Aerospace Sciences Meeting, Jan. 10–13, 2005, Reno, NV

Adjoint Formulation for an Embedded-Boundary Cartesian Method

Marian Nemec* and Michael J. Aftosmis†
NASA Ames Research Center, Moffett Field, CA 94035
Scott M. Murman‡
ELORET Corp., Moffett Field, CA 94035
Thomas H. Pulliam§
NASA Ames Research Center, Moffett Field, CA 94035

A discrete-adjoint formulation is presented for the three-dimensional Euler equations discretized on a Cartesian mesh with embedded boundaries. The solution algorithm for the adjoint and flow-sensitivity equations leverages the Runge–Kutta time-marching scheme in conjunction with the parallel multigrid method of the flow solver. The matrix-vector products associated with the linearization of the flow equations are computed on-the-fly, thereby minimizing the memory requirements of the algorithm at a computational cost roughly equivalent to a flow solution. Three-dimensional test cases, including a wing-body geometry at transonic flow conditions and an entry vehicle at supersonic flow conditions, are presented. These cases verify the accuracy of the linearization and demonstrate the efficiency and robustness of the adjoint algorithm for complex geometry problems.

I. Introduction

Adjoint solutions of the governing flow equations are becoming increasingly important for the development of efficient analysis and optimization algorithms. A well-known use of the adjoint method is gradient-based shape optimization.[1–5] Given an objective function that defines some measure of performance, such as the lift and drag functionals, its gradient is computed at a cost that is essentially independent of the number of design variables (e.g., geometric parameters that control the shape). Recent adjoint applications focus on the analysis problem, where the adjoint solution is used to drive mesh adaptation,[6–9] as well as to provide estimates of functional error bounds and corrections.[10] The attractive feature of this approach is that the mesh-adaptation procedure targets a specific functional, thereby localizing the mesh refinement and reducing computational cost.

The adjoint method, as well as the closely related flow-sensitivity (or direct) method, have been extensively analyzed and validated for the Euler and Navier–Stokes equations discretized on structured[11–14] and

*NRC Research Associate; currently Research Scientist, ELORET Corp., Applications Branch, MS T27B; mnemec@mail.arc.nasa.gov. Member AIAA.
†Research Scientist, maftosmis@mail.arc.nasa.gov. Senior Member AIAA.
‡Senior Research Scientist, smurman@mail.arc.nasa.gov. Member AIAA.
§Senior Research Scientist, tpulliam@mail.arc.nasa.gov. Associate Fellow AIAA.

unstructured meshes.[15–17] When the adjoint equation is derived from the discrete form of the flow equations, which is the approach used in this work, the accuracy of the adjoint solution primarily depends on the accuracy of the linearization of the flow equations. The most common approach is to derive the linearization by hand.[15,18–20] An emerging approach is automatic differentiation,[21–25] which aims to generate the linearization with minimal human effort. The solution of the adjoint equation is usually accomplished using the same scheme that solves the flow equations.

Cartesian methods provide a promising alternative to discretizations based on structured and unstructured meshes. The key feature of Cartesian methods is that the volume mesh generation is fast and robust for arbitrarily-complex geometry. In the embedded-boundary approach, the surface discretization is intersected against a Cartesian volume mesh resulting in a layer of arbitrary polyhedra, or cut cells, adjacent to the surface.[26,27] When combined with an efficient flow solver,[28] this approach is particularly well suited for the automated analysis of complex geometry problems, and consequently a promising approach for optimal shape design. Nemec *et al.*[29] presented a CAD-based optimization framework for an embedded-boundary Cartesian method where the objective function gradients are computed using finite differences. In addition, Aftosmis and Berger[30] presented an efficient strategy for h-refinement of nested Cartesian meshes. Consequently, an adjoint solver would enhance the efficiency of the existing optimization framework, and provide a new adaptation criterion to complement the present feature-detection and local truncation error approaches.

In previous work on Cartesian adjoint solvers, Melvin *et al.*[31] developed an adjoint formulation for the TRANAIR code,[32] which is based on the full-potential equation with viscous corrections. More recently, Dadone and Grossman[33,34] presented an adjoint formulation for the two-dimensional Euler equations using a ghost-cell method to enforce the wall boundary conditions.

In this paper, we develop an adjoint formulation for the three-dimensional Euler equations discretized on an embedded-boundary Cartesian mesh. We present a numerical implementation of the discrete adjoint approach, where the adjoint solver efficiently reuses the time-marching, multigrid, and domain decomposition schemes of the flow solver. We discuss a face-based algorithm for the construction of the flow Jacobian matrix that minimizes memory usage. The approach is generally applicable to finite-volume schemes for unstructured meshes and is tailored to the specialized data structures of the flow solver. We verify the accuracy of the linearization and demonstrate the overall efficiency and robustness of the approach on several two- and three-dimensional test cases.

II. Problem Formulation

The aerodynamic optimization problem we consider in this work consists of determining values of design variables X, such that the objective function $\mathcal{J}$ is minimized

$$\min_X \ \mathcal{J}(X, Q) \tag{1}$$

where the vector Q denotes the continuous, conservative flow variables. The flow variables are forced to satisfy the governing flow equations within a feasible region of the design space Ω

$$\mathcal{F}(X, Q) = 0 \qquad \forall\ X \in \Omega \tag{2}$$

which implicitly defines $Q = f(X)$.

Our goal is to solve the optimization problem defined by Eqs. 1–2 using a gradient-based method. The adjoint equation required for the computation of the gradient, $\mathrm{d}\mathcal{J}/\mathrm{d}X$, can be derived using either the continuous or discrete formulation. In the continuous formulation, the adjoint equation is derived directly from Eqs. 1–2, and the resulting continuous equation is then discretized. Alternatively, Eqs. 1–2 can be first discretized, and the desired adjoint equation is derived from the discrete form of the governing equations. Both formulations work well in practice. We favour the discrete formulation because it provides a systematic approach to the adjoint code development by closely following the flow solver. Verification of the resulting code is straightforward, and code maintenance issues such as the incorporation of new features

of the flow solver are simplified. In order to clearly present the discrete adjoint method, we first provide a brief description of the flow solver. Thereafter, we focus on the development of the gradient computation algorithm.

III. Governing Flow Equations and Numerical Method

The governing flow equations are the three-dimensional Euler equations of a perfect gas. For a finite region of space with volume V and surface area S, the integral form of the Euler equations is given by

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_V Q\ \mathrm{d}V + \oint_S \mathbf{F}\cdot n\ \mathrm{d}S = 0 \tag{3}$$

where $Q = [\rho, \rho u, \rho v, \rho w, \rho E]^{\mathrm{T}}$, $\mathbf{F}$ is the inviscid flux tensor and n is the outward facing unit normal vector.

A. Spatial Discretization

The Euler equations are solved with a finite-volume method on a regular Cartesian mesh with embedded boundaries. The mesh consists of hexahedral cells, except for a layer of body-intersecting cells, or cut-cells, that are arbitrary polyhedra adjacent to the boundaries.[27] Spatial discretization uses a cell-centered approach, where the control volumes V correspond to the mesh cells and the cell-averaged value of Q, denoted by $\bar{Q}$, is located at the centroid of each cell. The control volumes are fixed in time, resulting in the following semi-discrete form of Eq. 3:

$$\vec{V}\ \frac{\mathrm{d}\vec{Q}}{\mathrm{d}t} + \vec{R}\left(\vec{Q}\right) = 0 \tag{4}$$

where $\vec{Q} = [\bar{Q}_1, \bar{Q}_2, \ldots, \bar{Q}_N]^{\mathrm{T}}$ is the discrete solution vector for all N cells, $\vec{V}$ is a diagonal matrix containing the corresponding cell volumes, and $\vec{R}$ is the residual vector. The residual in each cell i is expressed as

$$R_i = \sum_{\mathrm{j}\in \mathrm{V_i}} \hat{\mathbf{F}}_j \cdot n_j S_j \tag{5}$$

where j denotes the jth face of volume V_i with area S and $\hat{\mathbf{F}}$ represents the numerical flux function. The flux function is evaluated at the face centroids using the flux-vector splitting approach of van Leer.[35]

The residual evaluation for a second-order accurate discretization proceeds by first reconstructing the solution to the cell face. This is illustrated in Fig. 1, for two neighbouring Cartesian cells i, k sharing a common face. Primitive variables, $U = [\rho, u, v, w, p]^{\mathrm{T}}$, are used for the reconstruction and the left and right states are given by

$$U_{\mathrm{L}} = \bar{U}_i + d_{\mathrm{L}}\phi_i \nabla U_i \tag{6}$$

$$U_{\mathrm{R}} = \bar{U}_k - d_{\mathrm{R}}\phi_k \nabla U_k \tag{7}$$

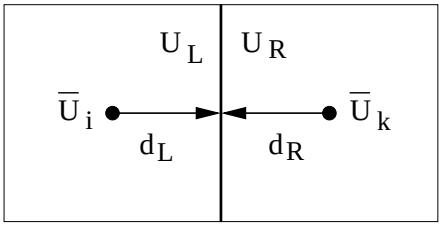


Figure 1. Labels for the reconstruction of cell-centroid values to a common face

where d_{L} and d_{R} are the distances from the cell centroids to the face centroid, ϕ is the slope limiter used to enforce monotonic solutions, and ∇U is the solution gradient determined via a linear least-squares procedure. Next, the evaluation of the van Leer flux function provides a unique value of flux at the face

$$\hat{\mathbf{F}}(U_{\mathrm{L}}, U_{\mathrm{R}}) = \mathbf{f}^{+}(U_{\mathrm{L}}) + \mathbf{f}^{-}(U_{\mathrm{R}}) \tag{8}$$

where the resulting flux is computed in conservative form. At the implementation level, the assembly of the residual vector is accomplished by a loop over the faces of the mesh using a specialized, faced-based data

structure.[28] The flux contributions are scattered from the face and accumulated in the cells

$$R_i = R_i + \hat{\mathbf{F}} S \tag{9}$$

$$R_k = R_k - \hat{\mathbf{F}} S \tag{10}$$

where the sign reflects the change in the direction of the outward-pointing normal, see Fig. 1. The boundary conditions are enforced weakly by appropriate modifications of the reconstructed state and the resulting flux.

B. Steady-State Solution Algorithm

Steady-state solutions are obtained using a five-stage Runge–Kutta scheme accelerated by local time stepping and multigrid. The multigrid residual restriction operator is the sum of the residuals of the fine mesh cells enclosed by the coarse cell, while the prolongation operator is direct injection. Convergence to steady state is further accelerated by a highly-scalable domain decomposition scheme. For further details on the flow solution algorithm, see Aftosmis *et al.*[28,36,37] and Berger *et al.*[37]

IV. Adjoints and Sensitivities

A. Formulation

The gradient, G, of the discrete objective function $\mathcal{J}\left[X, \vec{Q}(X)\right]$ is given by

$$G = \frac{\mathrm{d}\mathcal{J}}{\mathrm{d}X} = \frac{\partial \mathcal{J}}{\partial X} + \frac{\partial \mathcal{J}}{\partial \vec{Q}} \frac{\mathrm{d}\vec{Q}}{\mathrm{d}X} \tag{11}$$

where $\vec{Q}$ is the steady-state solution of Eq. 4 for a given set of design variables

$$\vec{R}(X, \vec{Q}) = 0 \tag{12}$$

Note that this is the discrete equivalent of Eq. 2. We reduce the vector of design variables, X, to a scalar in order to clearly distinguish between partial and total derivatives. For problems with multiple design variables, G and $\partial\mathcal{J}/\partial X$ are $[1 \times N_{\mathrm{D}}]$ row vectors, $\partial\mathcal{J}/\partial\vec{Q}$ is a $[1 \times N_{\mathrm{F}}]$ row vector, and $\mathrm{d}\vec{Q}/\mathrm{d}X$ is a $[N_{\mathrm{F}} \times N_{\mathrm{D}}]$ matrix, where N_{F} and N_{D} represent the number of flow and design variables, respectively.

In Eq. 11, the evaluation of the term $\mathrm{d}\vec{Q}/\mathrm{d}X$, referred to as the flow sensitivities, is obtained by differentiating Eq. 12 with respect to the design variables

$$\frac{\partial \vec{R}}{\partial \vec{Q}} \frac{\mathrm{d}\vec{Q}}{\mathrm{d}X} = -\frac{\partial \vec{R}}{\partial X} \tag{13}$$

We assume that the implicit function $\vec{Q}(X)$ is sufficiently smooth[5,38] and note that $\mathrm{d}\vec{R}/\mathrm{d}X = 0$. The direct, or flow-sensitivity, method results from solving Eq. 13 for the flow sensitivities $\mathrm{d}\vec{Q}/\mathrm{d}X$ and using these values in Eq. 11 to obtain the gradient.

In order to formulate the discrete-adjoint method, substitute Eq. 13 into Eq. 11 to obtain

$$\frac{\mathrm{d}\mathcal{J}}{\mathrm{d}X} = \frac{\partial \mathcal{J}}{\partial X} - \frac{\partial \mathcal{J}}{\partial \vec{Q}} \left(\frac{\partial \vec{R}}{\partial \vec{Q}}\right)^{-1} \frac{\partial \vec{R}}{\partial X} \tag{14}$$

From the triple-product term in Eq. 14, define the following intermediate problem involving the first two terms

$$\frac{\partial \vec{R}}{\partial \vec{Q}}^{\mathrm{T}} \vec{\psi} = \frac{\partial \mathcal{J}}{\partial \vec{Q}}^{\mathrm{T}} \tag{15}$$

where $\vec{\psi}$ is a $[N_{\mathrm{F}} \times 1]$ column vector. This linear system of equations is referred to as the adjoint equation, and the vector $\vec{\psi}$ represents the adjoint variables. Substituting $\vec{\psi}$ into Eq. 14, the expression for the gradient becomes

$$\frac{\mathrm{d}\mathcal{J}}{\mathrm{d}X} = \frac{\partial \mathcal{J}}{\partial X} - \vec{\psi}^{\,\mathrm{T}} \frac{\partial \vec{R}}{\partial X} \tag{16}$$

In general, the problem of solving the adjoint equation is as difficult as solving the flow-sensitivity equation. Equation 13 is a linear system with multiple right-hand sides dependent on the number of design variables, while Eq. 15 is a linear system with multiple right-hand sides dependent on the number of objectives. In a typical optimization problem where the number of design variables exceeds the number of objectives, the adjoint method is therefore more efficient when iterative solvers are used to solve the linear systems.

B. Numerical Implementation

The solution of the adjoint and flow-sensitivity equations proceeds by introducing unsteady terms in Eqs. 13 and 15 to obtain the following semi-discrete forms

$$\vec{V} \frac{\mathrm{d}\vec{Q}'}{\mathrm{d}t} + \vec{R}^{\mathrm{s}} = 0 \tag{17}$$

$$\vec{V} \frac{\mathrm{d}\vec{\psi}}{\mathrm{d}t} + \vec{R}^{\mathrm{a}} = 0 \tag{18}$$

where $\vec{Q}' = \mathrm{d}\vec{Q}/\mathrm{d}X$ and the flow-sensitivity and adjoint residual vectors are given by

$$\vec{R}^{\mathrm{s}} = \frac{\partial \vec{R}}{\partial \vec{Q}} \vec{Q}' + \frac{\partial \vec{R}}{\partial X} \tag{19}$$

$$\vec{R}^{\mathrm{a}} = \frac{\partial \vec{R}}{\partial \vec{Q}}^{\mathrm{T}} \vec{\psi} - \frac{\partial \mathcal{J}}{\partial \vec{Q}}^{\mathrm{T}} \tag{20}$$

We expect Eqs. 17 and 18 to have similar stability properties to the flow equations, Eq. 4, since the eigenvalues of the flow Jacobian matrix are not changed by the transpose operator.

A critical step in the solution procedure is the evaluation of the matrix-vector product terms

$$\frac{\partial \vec{R}}{\partial \vec{Q}} \vec{Q}' \quad \text{and} \quad \frac{\partial \vec{R}}{\partial \vec{Q}}^{\mathrm{T}} \vec{\psi}$$

For the flow-sensitivity equation, the linearization of the residual equations can be approximated with finite-differences, for example

$$\frac{\partial \vec{R}}{\partial \vec{Q}} \vec{Q}' = \frac{\vec{R}\left(\vec{Q} + \epsilon \vec{Q}'\right) - \vec{R}(\vec{Q})}{\epsilon} \tag{21}$$

where a good choice of the stepsize is $\epsilon = \sqrt{(\epsilon_{\mathrm{m}})}/||v||$.[14,39] We refer to the formulation resulting from Eq. 21 as matrix-free flow sensitivities. Advantages of this approach are low memory requirements and ease of implementation, since the differentiation of cumbersome residual functions, such as limiters and numerical fluxes, is "automatically" provided. Unfortunately, this method does not extend to approximating the product of the adjoint variables with the transpose of the flow-Jacobian.

Strategies for the computation of the matrix-vector products based on an explicit linearization of the residual equations are usually driven by the choice of the flow solution method. For strongly implicit solvers, the non-zero entries of the sparse flow-Jacobian matrix can be precomputed and stored, thereby reducing CPU time but increasing memory usage.[40] Alternatively, some or all entries of the flow-Jacobian can be

recomputed when forming the matrix-vector product, thereby reducing memory usage but increasing CPU time, for example see Giles *et al.*,[15] Nielsen *et al.*,[16] and Mavriplis.[17]

To adopt the explicit flow solution method outlined in Sec. III for the solution of the adjoint and flow-sensitivity equations, we implement a face-based algorithm to efficiently recompute the flow-Jacobian entries at each evaluation of the residual vector. The approach is similar to the work of Barth[18] and Giles *et al.*[15] The formation of the flow-sensitivity matrix-vector product is simply a linearization of the flow-solution procedure. The first step is the multiplication by the Jacobian of the transformation from conservative to primitive variables

$$\vec{U}' = \frac{\partial \vec{U}}{\partial \vec{Q}} \vec{Q}' \tag{22}$$

where $\partial \vec{U}/\partial \vec{Q}$ is a block diagonal matrix.

Next, we seek the matrix-vector product associated with the linearization of the reconstructed left and right states, Eqs. 6 and 7, which we symbolically denote as $(\partial U_{\mathrm{L/R}}/\partial U)\vec{U}'$. Let the operator $\mathcal{L}$ represent the reconstruction operator

$$U_{\mathrm{L/R}} = \mathcal{L}(\bar{U}_m, \ldots, \bar{U}_n) \tag{23}$$

where the reconstructed state depends *linearly* on a set of neighbouring cells $(\bar{U}_m, \ldots, \bar{U}_n)$ and contains the least-squares weights that are dependent only on mesh geometry. Note that the limiter, ϕ in Eq. 6, is treated as a constant during the linearization process. We investigate the validity of this assumption in Section V. Barth[18] demonstrates that the desired matrix-vector product can be expressed as

$$\frac{\partial U_{\mathrm{L/R}}}{\partial U} \vec{U}' = \mathcal{L}(\vec{U}') = U'_{\mathrm{L/R}} \tag{24}$$

Hence, the matrix-vector product associated with the linearization of the reconstruction operator is the reconstruction operator itself evaluated using the flow-sensitivity vector instead of the flow variables. For first-order spatial discretization, the linearization of the reconstruction operator is simply the identity matrix.

The final step is the matrix-vector product associated with the linearization of the van Leer flux function, Eq. 8. This flux function has been linearized previously, see for example Burgreen.[41] We derived and coded the linearization by hand and verified the result using the symbolic differentiation tool of Maple 6. We found that the Maple code executed roughly a factor of two slower than the hand-generated code. Referring to Fig. 1, the contribution to the flow-sensitivity residual vector from a common face to its left and right neighboring cells can be written as

$$R_i^{\mathrm{s}} = R_i^{\mathrm{s}} + \left(\frac{\partial \mathbf{f}^+}{\partial U_{\mathrm{L}}} U'_L + \frac{\partial \mathbf{f}^-}{\partial U_{\mathrm{R}}} U'_R \right) S \tag{25}$$

$$R_k^{\mathrm{s}} = R_k^{\mathrm{s}} - \left(\frac{\partial \mathbf{f}^+}{\partial U_{\mathrm{L}}} U'_L + \frac{\partial \mathbf{f}^-}{\partial U_{\mathrm{R}}} U'_R \right) S \tag{26}$$

For the adjoint matrix-vector product, the transpose operator reverses the order of matrix multiplication outlined for the flow-sensitivity procedure. Hence, the adjoint of the flux-function linearization is evaluated first, followed by adjoint of the reconstruction step, and lastly the adjoint of the transformation Jacobian form conservative to primitive variables. By examining the structure of the transposed flow Jacobian matrix for a first-order accurate spatial discretization, the contribution to the adjoint residual vector from a common face to its left and right neighboring cells, see Fig. 1, can be written as

$$R_i^{\mathrm{a}} = R_i^{\mathrm{a}} + \frac{\partial U_i}{\partial Q_i}^{\mathrm{T}} \frac{\partial \mathbf{f}^+}{\partial U_{\mathrm{i}}}^{\mathrm{T}} S \left(\psi_i - \psi_k \right) \tag{27}$$

$$R_k^{\mathrm{a}} = R_k^{\mathrm{a}} + \frac{\partial U_k}{\partial Q_k}^{\mathrm{T}} \frac{\partial \mathbf{f}^-}{\partial U_{\mathrm{k}}}^{\mathrm{T}} S \left(\psi_{\mathrm{i}} - \psi_{\mathrm{k}} \right) \tag{28}$$

where the gather operation $\psi_i - \psi_k$ reflects the change of sign in Eqs. 9 and 10. Cusdin[24] derives a similar result using automatic differentiation. For second-order spatial discretizations, the adjoint of the reconstruction procedure is accomplished using an additional pass through the faces of the mesh. The Jacobian of the reconstruction procedure involves only the geometry-dependent least-squares weights, which are already computed and stored by the flow solver.

Since the algorithm uses only elemental face operations for both the flow-sensitivity and adjoint matrix-vector products, the face-based data structures and the domain decomposition scheme of the flow solver are reused directly. The implementation results in only a slight increase in the memory usage over the flow solver, namely we require two additional arrays that store the converged flow solution and its gradient.

The partial derivative term $\partial\mathcal{J}/\partial\vec{Q}$ in Eqs. 11 and 15 is derived by hand. The differentiation assumes first-order spatial discretization, that is, we do not reconstruct the value of pressure to the surface. The remaining partial derivative terms in Eqs. 11 and 16, namely the objective function sensitivity $\partial\mathcal{J}/\partial X$ and the residual sensitivity $\partial\vec{R}/\partial X$, are approximated with finite differences. This approach works well for the verification exercises presented in the next section, however, it is not sufficient for general shape optimization problems since the topology of the Cartesian mesh may change as the shape is perturbed. This issue is left as a subject of future work.

Convergence to steady-state is accomplished using the same five-stage Runge–Kutta time marching and multigrid schemes of the flow solver. Giles[42] derived conditions for Runge–Kutta time marching schemes and multigrid that ensure the same convergence of the objective function gradient for the adjoint and flow-sensitivity methods. This duality-preserving algorithm is implemented almost automatically, since the existing residual prolongation operator is a transpose of the restriction operator. Overall, the CPU time per iteration of the adjoint and flow-sensitivity solvers is roughly equivalent to the flow solver. This is because the additional cost of re-evaluating the matrix vector product at each iteration is offset by preprocessing the local time step and limiter values, as well as not requiring positivity checks for solution updates.

V. Results and Discussion

Several two- and three-dimensional test cases are studied to verify the accuracy and characterize the performance of the adjoint and flow-sensitivity solvers. Finite-difference gradients provide a benchmark that is used to establish the accuracy of the linearization. The relative finite-difference stepsize (ϵ) is varied from 1% to 0.01% to ensure accurate gradient estimates.

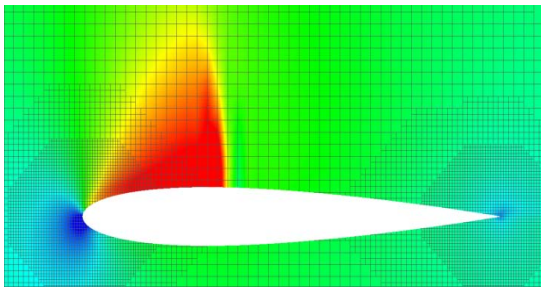
A. NACA 0012 Airfoil

The first test case involves the NACA 0012 airfoil at transonic flow conditions. Our main goal is to verify the linearization of the spatial discretization and investigate the effect of the constant limiter assumption. The freestream Mach number is 0.7 and the angle of incidence is 3 deg. The two-dimensional mesh contains roughly $11,000$ cells. For this test case, the multigrid and domain decomposition algorithms are not used, and the flow-solution gradients are computed at every stage of the five-stage Runge–Kutta (RK5) time-marching scheme. The CFL number is 1.3. The Mach number distribution and the computational mesh are shown in Fig. 2(a).

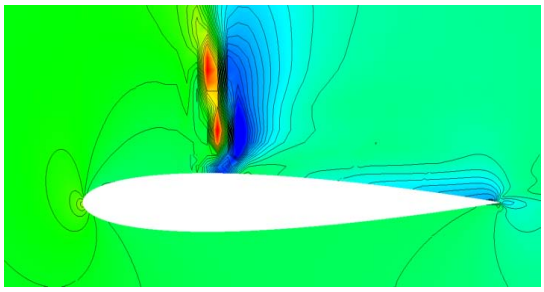
The objective function is $-C_L$, which represents a lift maximization problem. We compute the gradient of the objective function with respect to two design variables. The first design variable is a local shape perturbation on the upper surface of the airfoil at $x = 33.5\%c$. This location is just upstream of the shock in Fig. 2(a). We modify the airfoil shape in a single cell only, which avoids changes in mesh topology. The second design variable is the angle of incidence. For the computation of the gradient with respect to the first design variable, the limiter function (ϕ in Eq. 6) is set to one, i.e. the limiter is not used.

Figures 2(b) and 2(c) show the qualitative behaviour of the flow-sensitivity and adjoint variables corresponding to the y-momentum equation, respectively. The flow-sensitivities attain large values and are discontinuous at the shock, and have much smaller values near the leading and trailing edges of the airfoil.

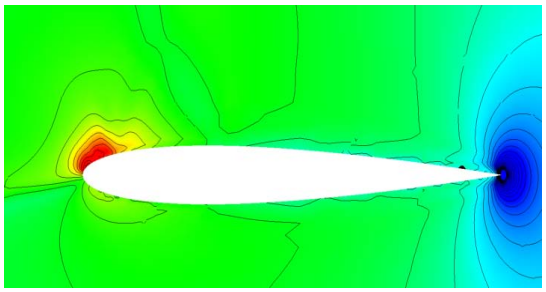
These results are consistent with the location of the design variable near the base of the shock. The adjoint variables represent the sensitivity of the functional, in this case $-C_{\mathrm{L}}$, to a unit perturbation of the residual equations. The main feature is a singularity at the trailing edge, emphasizing the high sensitivity of lift to perturbations in the y-momentum equation in this area. Furthermore, the adjoint variables are continuous across the shock and increase again in the supersonic region near the leading edge. A weaker structure propagates upstream from the leading-edge stagnation point.



(a) Mach number (red $>$ 1.2, blue $<$ 0.4)



(b) Flow-sensitivity contours for $\partial(\rho v)/\partial y$ (red $>$ 1.8, blue $<$ -1.8)

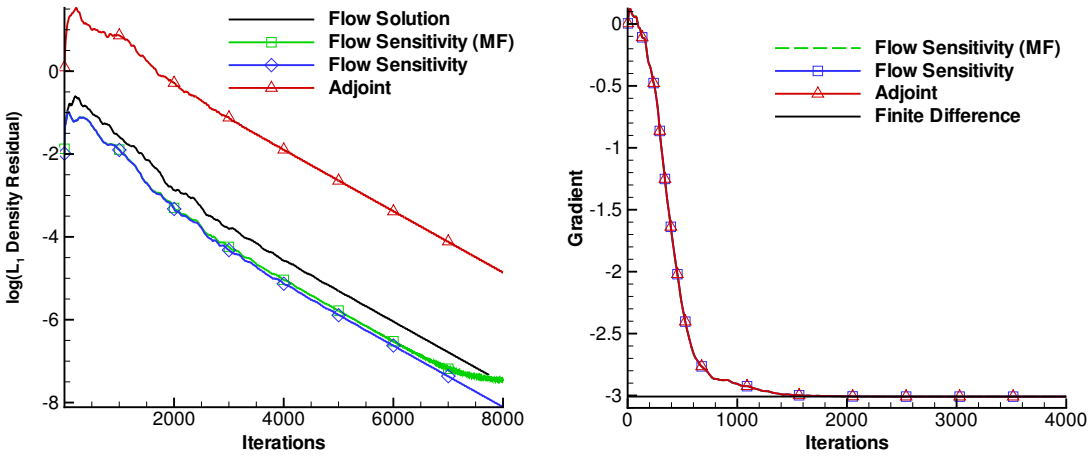


(c) Adjoint contours of the y-momentum equation for the lift functional (red $>$ 100, blue $<$ -100)

Figure 2. NACA 0012 test case ($M_\infty = 0.7$, $\alpha = 3$ deg.)

Figure 3(a) shows the convergence history of the flow, flow-sensitivity, and adjoint equations. The symbol MF denotes the matrix-free approach based on Eq. 21. The asymptotic convergence rate is the same for all equations. The matrix-free approach stalls due to the choice of finite-difference stepsize and occurs near the single-precision limit. Figure 3(b) shows the convergence of the objective function gradient. The agreement between the flow-sensitivity, adjoint, and finite-difference methods is excellent at convergence. Furthermore, the adjoint and flow-sensitivity methods generate identical values of the gradient at each iteration. For this case, this equivalence between the methods is expected since the flow-solution gradients are updated at each stage of the RK5 scheme.

In Fig. 4, we verify the gradient accuracy with respect to the angle of incidence design variable. This case is more sensitive to the selection of the finite-difference stepsize (ϵ), which is attributed to non-smooth changes in the shock position. We show two reference gradient values. For this case, the limiter function is used in order to investigate its effect on the accuracy of the gradient. Recall that the limiter is treated as a constant during the linearization except in the matrix-free approach. The differences in gradient values between the matrix-free and explicit linearization approaches are small and are bracketed within the finite-



(a) Residual (b) Gradient $\mathrm{d}(-C_\mathrm{L})/\mathrm{d}y$

Figure 3. Convergence histories for the NACA 0012 airfoil

difference values. This indicates that the error in neglecting the linearization of the limiter is small.

B. M100 Configuration

The second test case involves the M100 wing-body configuration[43] at transonic flow conditions. This test case provides a three-dimensional verification exercise for the accuracy of the linearization. We also examine the convergence of the adjoint and flow-sensitivity solvers when using the full solution algorithm, including partial updates of the RK5 scheme, multigrid, and parallel computing.

The freestream Mach number is 0.8027 and the angle of incidence is 2.873 deg. A half-body mesh is used, which contains roughly $1,926,000$ cells. Convergence to steady-state is achieved using 64 processors, a 4-level W-cycle multigrid with one pre- and one post-smoothing pass, and a CFL number of 1.3. Partial updates of the flow gradients are also used, i.e. the flow gradients are updated only on the first stage of the RK5 scheme. The Mach number distribution is shown in Fig. 5, where a strong shock is visible on the upper surface of the wing.

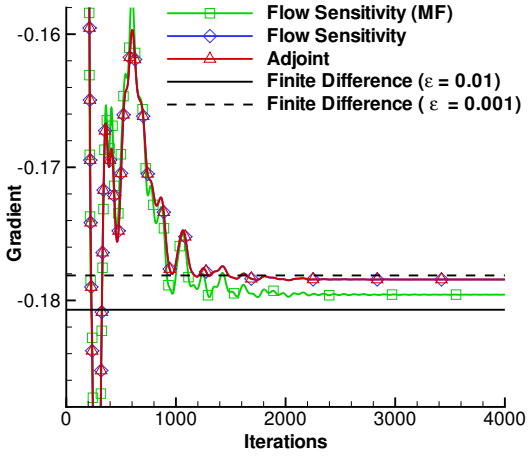


Figure 4. Effect of limiters on gradient accuracy

The objective function is $-C_\mathrm{L}$ and the design variable is the angle of incidence. The first verification exercise is performed using first-order accurate spatial discretization. Figure 6(a) shows the convergence of the flow, flow-sensitivity, and adjoint equations. Again, the asymptotic convergence rate is the same for all equations. The convergence of the objective function gradient is shown in Fig. 6(b), where the agreement with the finite-difference gradient is excellent. Note that the adjoint and flow-sensitivity solvers compute the same gradient values throughout the solution process. Also shown in Fig. 6(b) is the convergence of the objective

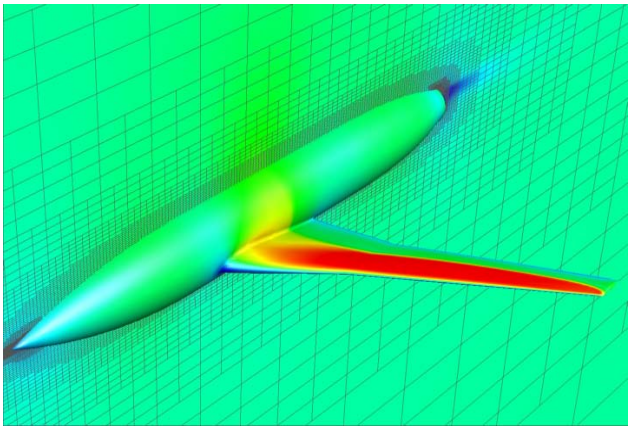
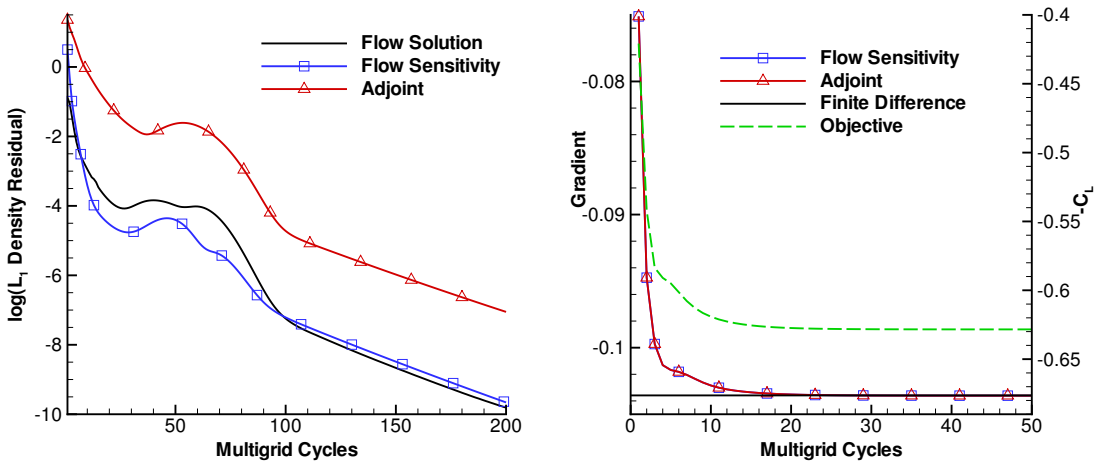


Figure 5. Mach contours for the M100 configuration (red > 1.4, blue < 0.4, $M_\infty = 0.8027$, $\alpha = 2.873$ deg.)

function. The objective and its gradient are converged within 20 multigrid cycles.



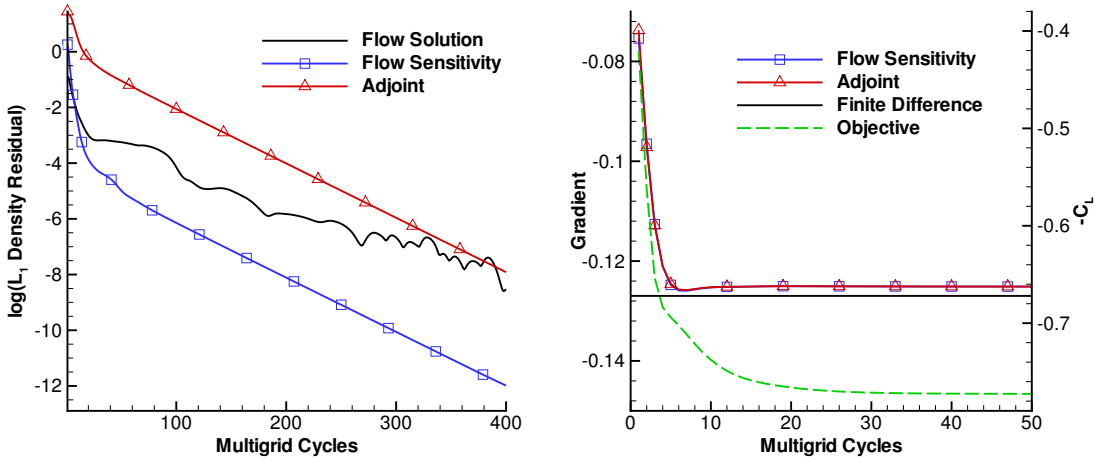
(a) Residual

(b) Gradient $\mathrm{d}(-C_\mathrm{L})/\mathrm{d}\alpha$ and objective function

Figure 6. Convergence histories for first-order accurate spatial discretization (M100 wing-body configuration, $M_\infty = 0.8027$, $\alpha = 2.873$ deg.)

We perform the same study using second-order spatial discretization and the results are shown in Fig. 7. Although the flow solver converges almost eight orders of magnitude, convergence is not smooth, as shown in Fig. 7(a). Nevertheless, the asymptotic convergence of the adjoint and flow-sensitivity equations is equivalent. Convergence of the objective and its gradient is shown in Fig. 7(b). Overall, the agreement between the finite-difference, adjoint, and flow sensitivity gradients is good. The small difference can be attributed to the constant limiter assumption during the linearization process, and we also observed that the value of the finite-difference gradient varied with the stepsize. We show the gradient value using $\epsilon = 0.005$. Lastly, close examination of Fig. 7(b) reveals slight differences in the gradient values between the adjoint and flow-sensitivity solvers during the convergence process. These differences are due to omitting an update of the

flow-solution gradient just prior to residual restriction, which is done to save overall CPU time, and the differences vanish at convergence.



(a) Residual

(b) Gradient $\mathrm{d}(-C_\mathrm{L})/\mathrm{d}\alpha$ and objective function

Figure 7. Convergence histories for second-order accurate spatial discretization (M100 wing-body configuration, $M_\infty = 0.8027$, $\alpha = 2.873$ deg.)

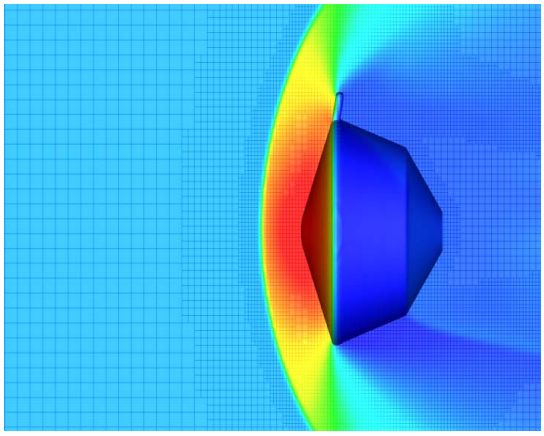
C. Mars Smart Lander

The final test case involves the Mars Smart Lander (MSL) entry vehicle[44] at supersonic flow conditions. The purpose of this verification exercise is to demonstrate the robustness of the adjoint and flow-sensitivity solvers for cases with complex flowfields. The MSL vehicle represents a real-life design study well suited for the application of new optimization algorithms. An interesting feature of the vehicle geometry is a small tab, visible near the top of the configuration in Fig. 8(a). This tab is used to alter the trim orientation (by changing the lift-to-drag ratio) to provide trajectory control.

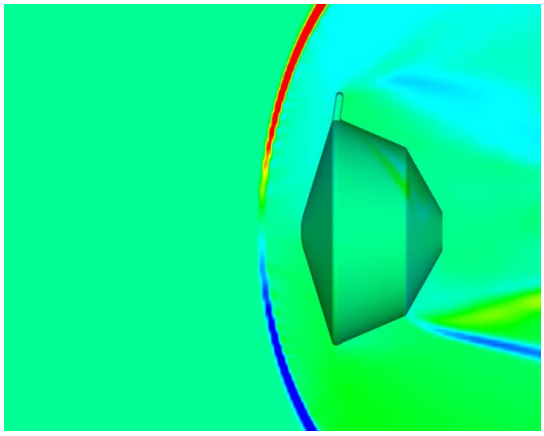
The freestream Mach number is 2.5 and the angle of incidence is 0 deg. The mesh contains roughly $1,770,000$ cells and was generated using the solution-adaptive h-refinement strategy presented in Ref. 30. Undivided differences of density were used to drive the adaptation procedure. A 3-level W-cycle multigrid with one pre- and one post-smoothing pass with a CFL number of 1.3 and 64 processors are used. The flow gradients are updated on the first stage of the RK5 scheme. The mesh and the flow solution (pressure contours) are shown in Fig. 8(a). The main flow features include a strong bow shock and an "unsteady" wake.

The objective function and design variable are again $-C_\mathrm{L}$ and the angle of incidence, respectively. Figures 8(b) and 8(c) show the contours of flow-sensitivities and adjoint variables for the energy equation. The sensitivity of the flow variables to angle of incidence perturbations is largest in the bow-shock region. The adjoint variables are continuous across the bow-shock and attain the largest values on the lower surface of the vehicle. Note the moderate adjoint values just upstream of the shock, highlighting the effect of the bow-shock orientation on the lift functional.

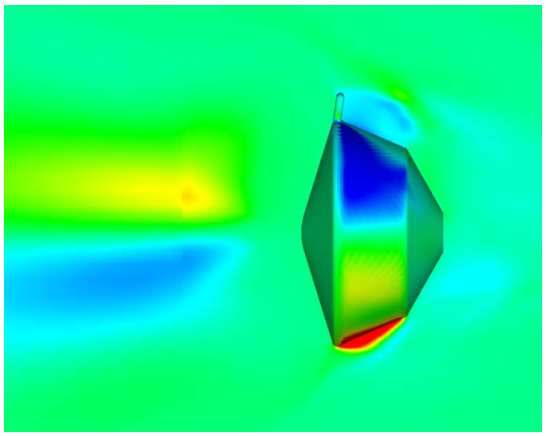
Convergence of the flow, flow-sensitivity, and adjoint equations are shown in Fig. 9. The flow-solution converged roughly four orders of magnitude and stalled. The stall in convergence is attributed to the unsteadiness of the wake. Full-multigrid startup is used, where the solution on the coarse grids (roughly



(a) Pressure contours (red > 5.5, blue < 0.1)



(b) Flow-sensitivity contours for $\partial(\rho E)/\partial\alpha$ (red $>$ 1, blue < -1)



(c) Adjoint contours of the energy equation for the lift functional (red > 0.04, blue < -0.02)

Figure 8. Mars Smart Lander test case ($M_\infty = 2.5$, $\alpha = 0$ deg.)

the first 100 multigrid cycles in Fig. 9) is very fast. Even with the limited convergence of the flow solution, the adjoint and flow-sensitivity solvers converge well. The asymptotic convergence rate of the adjoint and flow-sensitivity equations is the same. Objective function gradient accuracy is presented in Table 1. The agreement in Table 1 is good, and again the small differences are due to the constant limiter assumption in the linearization, and the dependence of the finite-difference gradient on the stepsize.

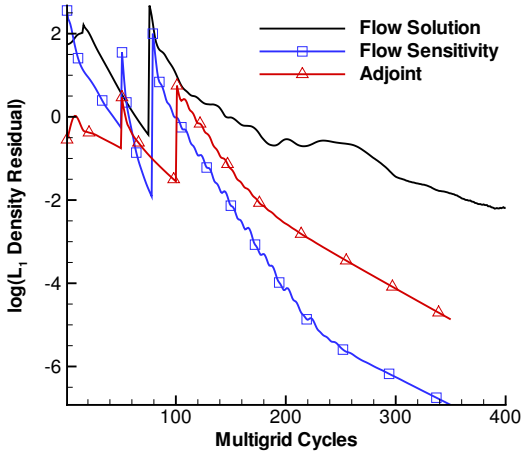


Figure 9. Residual convergence history for the Mars Smart Lander (3-level multigrid with full-multigrid start-up, $M_\infty = 2.5$, $\alpha = 0$ deg.)

Table 1. Gradient accuracy for the Mars Smart Lander ($\mathrm{d}(-C_\mathrm{L})/\mathrm{d}\alpha$)

Finite Difference[a]	Flow Sensitivity	Adjoint
0.02796	0.02589	0.02589

[a] Stepsize 0.005

VI. Summary and Future Work

Discrete adjoint and flow-sensitivity algorithms have been presented for the three-dimensional Euler equations discretized on an embedded-boundary Cartesian mesh. The algorithms reuse the time-marching and parallel multigrid methods of the flow solver and recompute the required matrix-vector products at each iteration. This approach results in memory usage and CPU cost roughly equivalent to the flow solver. We verified the accuracy of the gradient computation by comparison with finite-difference approximations and obtained excellent agreement for both the adjoint and flow-sensitivity methods. Robust performance of the algorithms was demonstrated for all test cases.

Future work focuses on incorporating the adjoint solver into the CAD-based optimization framework of Ref. 29. The challenging aspect of this work is to accurately linearize the objective function and residual equations with respect to the design variables. However, the combination of an embedded-boundary Cartesian method for dealing with large shape changes with the efficiency of adjoint-based optimization techniques appears to be a promising approach for practical design problems.

VII. Acknowledgments

The funding of the first author by the National Research Council Research Associateship Award is gratefully acknowledged. Scott Murman was supported by NASA Ames Research Center (contract NAS2-00062) during this work.

References

[1]Jameson, A., "Aerodynamic Design via Control Theory," *Journal of Scientific Computing*, Vol. 3, 1988, pp. 233–260, Also ICASE report 88–64.

[2]Baysal, O. and Eleshaky, M. E., "Aerodynamic Sensitivity Analysis Methods for the Compressible Euler Equations," *Journal of Fluids Engineering*, Vol. 113, No. 4, 1991, pp. 681–688.

[3]Reuther, J. J., *Aerodynamic Shape Optimization Using Control Theory*, Ph.D. thesis, University of California Davis, 1996.

[4]Anderson, W. K. and Venkatakrishnan, V., "Aerodynamic Design Optimization on Unstructured Grids with a Continuous Adjoint Formulation," *Computers & Fluids*, Vol. 28, 1999, pp. 443–480.

[5]Giles, M. B. and Pierce, N. A., "An Introduction to the Adjoint Approach to Design," *Flow, Turbulence and Combustion*, Vol. 65, No. 3/4, 2000, pp. 393–415.

[6]Pierce, N. A. and Giles, M. B., "Adjoint Recovery of Superconvergent Functionals from PDE Approximations," *SIAM Review*, Vol. 42, No. 2, 2000, pp. 247–264.

[7]Venditti, D. A. and Darmofal, D. L., "Anisotropic Grid Adaptation for Functional Outputs: Application to Two-Dimensional Viscous Flow," *Journal of Computational Physics*, Vol. 187, 2003, pp. 22–46.

[8]Barth, T. J., "Numerical Methods and Error Estimation for Conservation Laws on Structured and Unstructured Meshes," Lecture notes, von Karman Institute for Fluid Dynamics, Series: 2003-04, Brussels, Belgium, March 2003.

[9]Park, M. A., "Adjoint-Based, Three Dimensional Error Prediction and Grid Adaptation," *AIAA Journal*, Vol. 42, No. 9, 2004, pp. 1854–1862.

[10]Pierce, N. A. and Giles, M. B., "Adjoint and defect error bounding and correction for functional estimates," *Journal of Computational Physics*, Vol. 200, 2004, pp. 769–794.

[11]Korivi, V. M., Taylor III, A. C., Newman, P. A., Hou, G. W., and Jones, H. E., "An Approximately Factored Incremental Strategy for Calculating Consistent Discrete Aerodynamic Sensitivity Derivatives," *Journal of Computational Physics*, Vol. 113, No. 2, 1994, pp. 336–346.

[12]Elliott, J. and Herling, W., "A Chimera Approach to Aerodynamic Shape Optimization for the Compressible, High-Re Navier Stokes Equations," AIAA Paper 2000–4729, Long Beach, CA, Sept. 2000.

[13]Kim, C. S., Kim, C., and Rho, O. H., "Sensitivity Analysis for the Navier–Stokes Equations with Two-Equation Turbulence Models," *AIAA Journal*, Vol. 39, No. 5, 2001, pp. 838–845.

[14]Nemec, M. and Zingg, D. W., "Newton–Krylov Algorithm for Aerodynamic Design Using the Navier–Stokes Equations," *AIAA Journal*, Vol. 40, No. 6, 2002, pp. 1146–1154.

[15]Giles, M. B., Duta, M. C., Müller, J.-D., and Pierce, N. A., "Algorithm Developments for Discrete Adjoint Methods," *AIAA Journal*, Vol. 41, No. 2, 2003, pp. 198–204.

[16]Nielsen, E. J., Lu, J., Park, M. A., and Darmofal, D. L., "An Implicit, Exact Dual Adjoint Solution Method for Turbulent Flows on Unstructured Grids," *Computers & Fluids*, Vol. 33, 2004, pp. 1131–1155.

[17]Mavriplis, D. J., "Formulation and Multigrid Solution of the Discrete Adjoint Problem on Unstructured Meshes," *ICCFD 3*, July 2004.

[18]Barth, T. J., "Parallel CFD Algorithms on Unstructured Meshes," *Special Course on Parallel Computing in CFD*, AGARD-R-807, Oct. 1995, pp. 7–1–7–41.

[19]Anderson, W. K. and Bonhaus, D. L., "Airfoil Design on Unstructured Grids for Turbulent Flows," *AIAA Journal*, Vol. 37, No. 2, 1999, pp. 185–191.

[20]Elliott, J. K., *Aerodynamic Optimization Based on the Euler and Navier–Stokes Equations Using Unstructured Grids*, Ph.D. thesis, Massachusetts Institute of Technology, 1998.

[21]Giering, R. and Kaminski, T., "Recipes for Adjoint Code Construction," *ACM Transactions on Mathematical Software*, Vol. 24, No. 4, 1998, pp. 437–474.

[22]Mohammadi, B. and Pironneau, O., *Applied Shape Optimization for Fluids*, Oxford University Press, New York, USA, 2001.

[23]Taylor III, A. C., Green, L. L., Newman, P. A., and Putko, M. M., "Some Advanced Concepts in Discrete Aerodynamic Sensitivity Analysis," *AIAA Journal*, Vol. 41, No. 7, 2003, pp. 1224–1229.

[24]Cusdin, P. and Müller, J.-D., "Deriving Linear and Adjoint Codes for CFD using Automatic Differentiation," Preprint submitted to AIAA J., www.ea.qub.ac.uk/jmueller/.

[25]Cusdin, P. and Müller, J.-D., "Generating Efficient Code with Automatic Differentiation," *4th European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS)*, edited by P. Neittaanmäki, T. Rossi, K. Majava, and O. Pironneau, July 2004.

[26]Aftosmis, M. J., "Solution Adaptive Cartesian Grid Methods for Aerodynamic Flows with Complex Geometries," Lecture notes, von Karman Institute for Fluid Dynamics, Series: 1997-02, Brussels, Belgium, March 1997.

[27]Aftosmis, M. J., Berger, M. J., and Melton, J. E., "Robust and Efficient Cartesian Mesh Generation for Component-Based Geometry," *AIAA Journal*, Vol. 36, No. 6, 1998, pp. 952–960.

[28]Aftosmis, M. J., Berger, M. J., and Adomavicius, G., "A Parallel Multilevel Method for Adaptively Refined Cartesian Grids with Embedded Boundaries," AIAA Paper 2000–0808, Reno, NV, Jan. 2000.

[29]Nemec, M., Aftosmis, M. J., and Pulliam, T. H., "CAD-Based Aerodynamic Design of Complex Configurations Using a Cartesian Method," AIAA Paper 2004–0113, Reno, NV, Jan. 2004.

[30]Aftosmis, M. J. and Berger, M. J., "Multilevel Error Estimation and Adaptive h-Refinement for Cartesian Meshes with Embedded Boundaries," AIAA Paper 2002–0863, Reno, NV, Jan. 2002.

[31]Melvin, R. G., Huffman, W. P., Young, D. P., Johnson, F. T., Hilmes, C. L., and Bieterman, M. B., "Recent Progress in Aerodynamic Design Optimization," *International Journal for Numerical Methods in Fluids*, Vol. 30, 1999, pp. 205–216.

[32]Young, D. P., Melvin, R. G., Bieterman, M. B., Johnson, F. T., and Samant, S. S., "A Locally Refined Rectangular Grid Finite Element Method: Application to Computational Fluid Dynamics and Computational Physics," *Journal of Computational Physics*, Vol. 92, No. 1, 1991, pp. 1–66.

[33]Dadone, A. and Grossman, B., "Efficient Fluid Dynamic Design Optimization Using Cartesian Grids," AIAA Paper 2003–3959, Orlando, FL, June 2003.

[34]Dadone, A. and Grossman, B., "Ghost-Cell Method for Inviscid Two-Dimensional Flows on Cartesian Grids," *AIAA Journal*, Vol. 42, No. 12, 2004, pp. 2499–2507.

[35]van Leer, B., "Flux-Vector Splitting for the Euler Equations," ICASE Report 82-30, Sept. 1982.

[36]Aftosmis, M. J., Berger, M. J., and Murman, S. M., "Applications of Space-Filling-Curves to Cartesian Methods for CFD," AIAA Paper 2004–1232, Reno, NV, Jan. 2004.

[37]Berger, M. J., Aftosmis, M. J., and Murman, S. M., "Analysis of Slope Limiters on Irregular Grids," AIAA Paper 2005–0490, Reno, NV, Jan. 2005.

[38]Giles, M. B. and Pierce, N. A., "Analytic Adjoint Solutions for the Quasi-One-Dimensional Euler Equations," *Journal of Fluid Mechanics*, Vol. 426, 2001, pp. 327–345.

[39]Nielsen, E. J., Anderson, W. K., Walters, R. W., and Keyes, D. E., "Application of Newton–Krylov Methodology to a Three-Dimensional Unstructured Euler Code," AIAA Paper 95–1733-CP, 1995.

[40]Nemec, M., *Optimal Shape Design of Aerodynamic Configurations: A Newton–Krylov Approach*, Ph.D. thesis, University of Toronto, 2003, http://oddjob.utias.utoronto.ca/marian/.

[41]Burgreen, G. W., *Three-Dimensional Aerodynamic Shape Optimization Using Discrete Sensitivity Analysis*, Ph.D. thesis, Old Dominion University, 1994.

[42]Giles, M. B., "On the Use of Runge–Kutta Time-Marching and Multigrid for the Solution of Steady Adjoint equations," Oxford University Computing Laboratory 00/10, June 2000.

[43]Carr, M. P. and Pallister, K. C., "Pressure Distributions Measured on Research Wing M100 Mounted on an Axisymmetric Body," *Experimental Data Base for Computer Program Assessment*, AGARD-R-138 B8, May 1979.

[44]Brown, J., Yates, L., Bogdanoff, D., Chapman, G., Loomis, M., and Tam, T., "Free Flight Testing in Support of the Mars Smart Lander Aerodynamics Database," AIAA Paper 2002–4410 (Rev. JSR), 2002.